MEMORIA PRÁCTICA 3

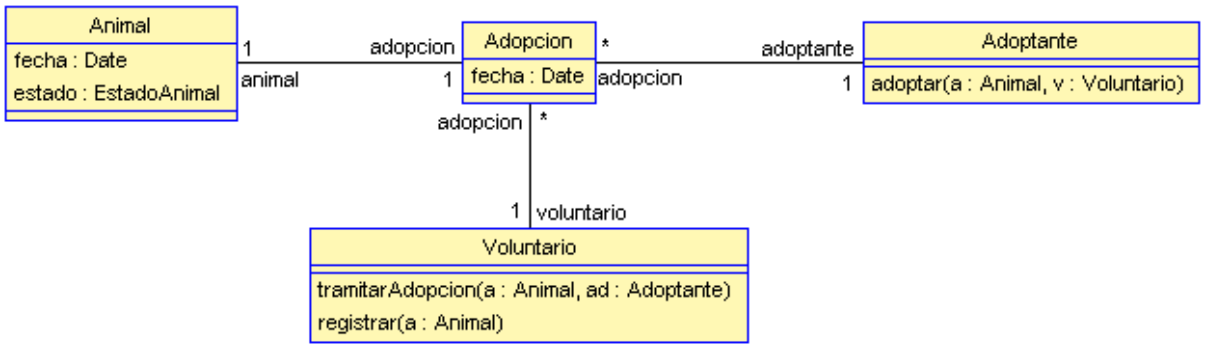
GR2-6

Castaños Benedicto, David
Cueto Díaz, Alejandro
Fernández Díaz, Oscar
Fernández González, Ángela
Labella Ramírez, Miguel
Rosales Santiago, Lucía

ÍNDICE

1. Apartado A

Para el diseño del código de andamiaje, hemos realizado la reificación de la clase asociación Adopción.



A la clase Adopción, además de su atributo fecha, se le añaden instancias de las clases Animal, Voluntario y Adoptante.

A la clase Animal se incorpora la instancia de la clase Adopción. En el caso de Adoptante se añade una lista de adopciones que contendrá todas las adopciones que haya realizado y podrá estar vacía. La clase Voluntario por su parte, tiene una lista de adopciones.

MOVERSE A —-----> CÓDIGO DEL EJERCICIO 1

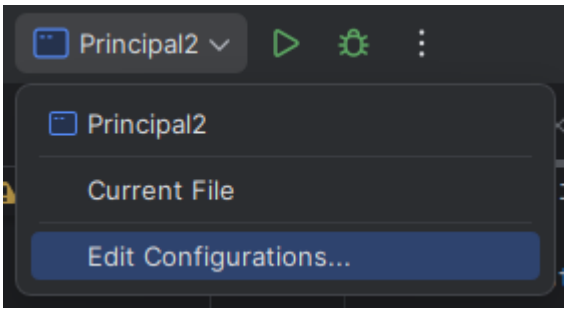
2. Apartado B

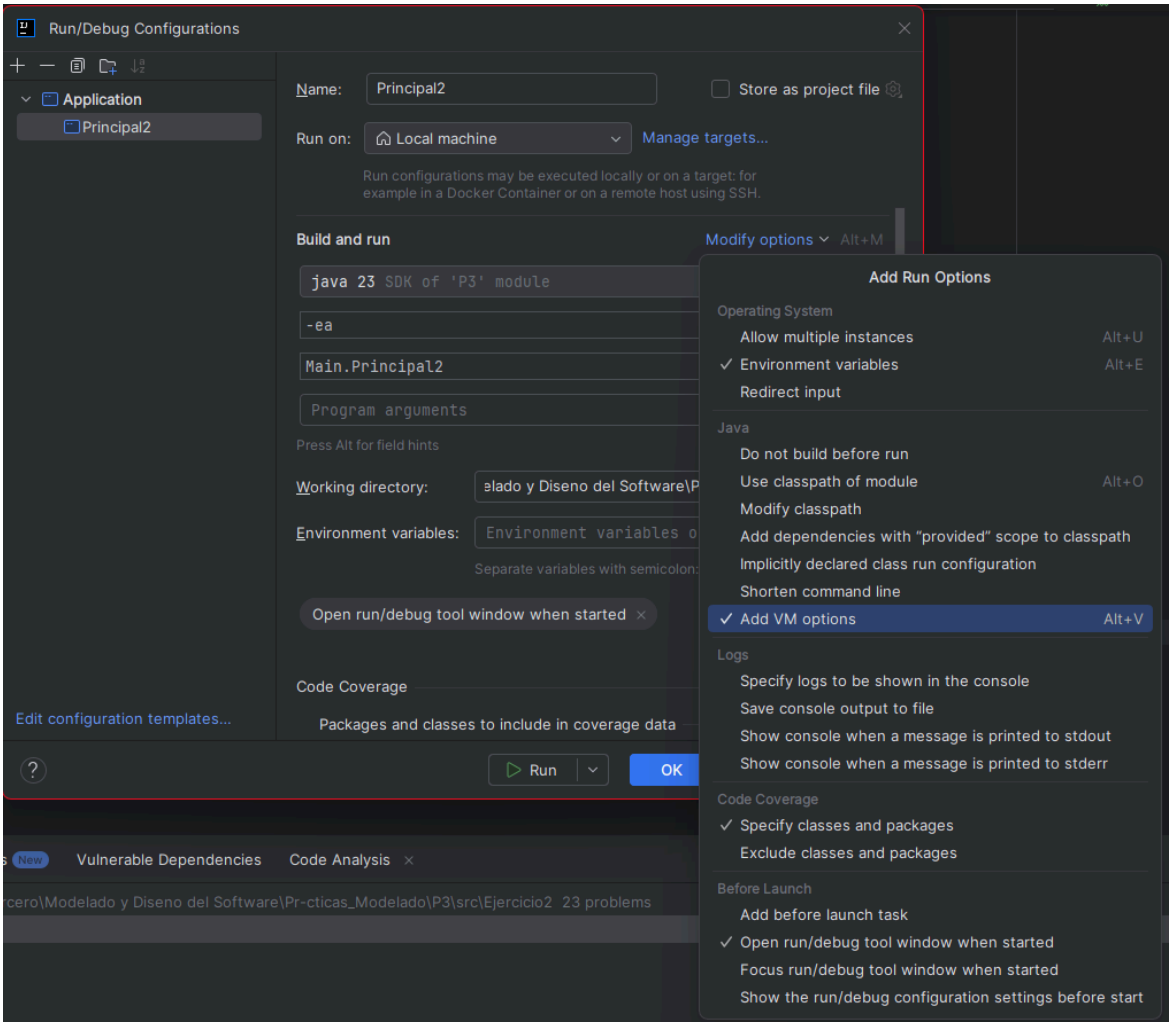
No se puede realizar una implementación directa en Java ya que una misma clase no puede recibir herencia múltiple por tanto hemos hecho uso de las interfaces para emular la herencia múltiple de manera controlada haciendo que un “Socio” pueda actuar como diferentes roles.

3. Apartado C

Como hemos explicado anteriormente hemos adaptado la implementación del ejercicio 1 con interfaces (Donante, Voluntario y Adoptante), y hemos mantenido las anteriores clases Donante, Voluntario y Adoptante bajo el nombre de RolDonante, RolVoluntario y RolAdoptante, eso sí añadiendo el atributo Socio a cada una de ellas que servirá para atribuir el “Rol” que el “Socio” quiera tener. El cambio más significativo se puede observar en la clase Socio en la cual hemos añadido tres atributos (voluntario, donante y adoptante) los cuales adjudican el “Rol” seleccionado.

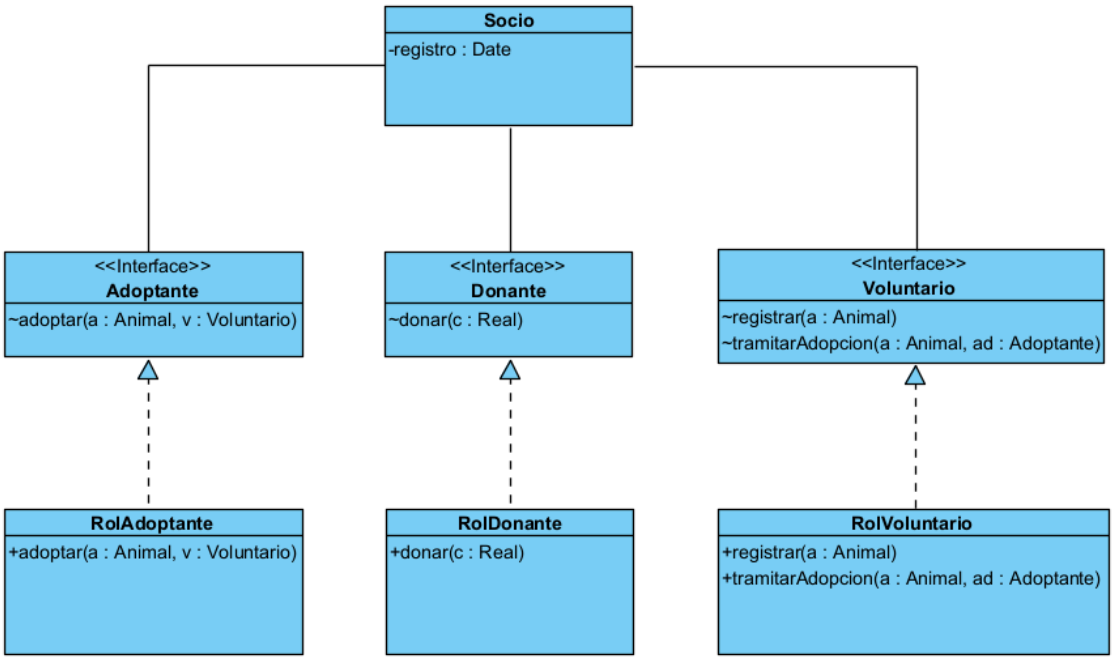
Observación: para ejecutar el código con control de errores debemos activar la opción Add VM options con el parámetro “-ea” en la configuración de la ejecución como se muestra a continuación:





MOVERSE A —-----> CÓDIGO DEL EJERCICIO 2

4. Apartado D



CÓDIGO EJERCICIO 1

```
package Ejercicio2;

import java.util.*;

public class RolAdoptante implements Adoptante {
   private Socio socio;
   private List<Adopcion> adopciones;

   public RolAdoptante(Socio socio) {
       setSocio(socio);
       this.adopciones = new ArrayList<>();
   }

   @Override
   public void adoptar(Animal animal, Voluntario voluntario) {
       assert (animal != null) : "El animal no es válido";
       assert (voluntario != null) : "El voluntario no es válido";

       Date fechaAdopcion = new Date();
       Adopcion adopcion = new Adopcion(fechaAdopcion, animal,
this, voluntario);
```

```
        assert (!adopciones.contains(adopcion)) : "Ya existe esta
adopción";

        for (Adopcion ad : adopciones) {
            assert (ad.noDup(adopcion)) : "Ya existe una adopción
para este animal";
        }

        addAdopcion(adopcion);
        animal.setAdopcion(adopcion);
        voluntario.tramitarAdopcion(animal, this);
    }

    @Override
    public Enumeration<Adopcion> getAdopciones() {
        return Collections.enumeration(adopciones);
    }

    @Override
    public void addAdopcion(Adopcion adopcion) {
        assert (adopcion != null) : "La adopción no es válida";
        assert (!adopciones.contains(adopcion)) : "Ya existe esta
adopción";
        for (Adopcion ad : adopciones) {
            assert (ad.noDup(adopcion)) : "Ya existe una adopción
para este animal";
        }
        adopciones.add(adopcion);
    }

    public Socio getSocio() {
        return socio;
    }

    public void setSocio(Socio socio) {
        assert (socio != null) : "El socio no es válido";
        this.socio = socio;
    }
}
```

```
package Ejercicio1;

import java.util.*;

public class Adoptante extends Socio {
    private List<Adopcion> adopciones;
```

```
    public Adoptante(Date registro, Refugio refugio) {
        super(registro, refugio);
        adopciones = new ArrayList<>();
    }

    public void adoptar(Animal animal, Voluntario voluntario) {
        assert (animal != null) : "El animal no es válido";
        assert (voluntario != null) : "El voluntario no es válido";

        Date fechaAdopcion = new Date();
        Adopcion adopcion = new Adopcion(fechaAdopcion, animal,
this, voluntario);

        assert (!adopciones.contains(adopcion)) : "Ya existe esta
adopción";

        for (Adopcion ad : adopciones) {
            assert (ad.noDup(adopcion)) : "Ya existe una adopción
para este animal";
        }

        addAdopcion(adopcion);
        animal.setAdopcion(adopcion);
        voluntario.tramitarAdopcion(animal, this);
    }

    public Enumeration<Adopcion> getAdopciones() {
        return Collections.enumeration(adopciones);
    }

    protected void addAdopcion(Adopcion adopcion) {
        assert (adopcion != null) : "La adopción no es válida";
        assert (!adopciones.contains(adopcion)) : "Ya existe esta
adopción";
        for (Adopcion ad : adopciones) {
            assert (ad.noDup(adopcion)) : "Ya existe una adopción
para este animal";
        }
        adopciones.add(adopcion);
    }
}
```

```
package Ejercicio1;

import java.util.Date;

public class Animal {
    private final Date nacimiento;
```

```
    private EstadoAnimal estado;
    private Refugio refugio;
    private Adopcion adopcion;

    public Animal(Date nacimiento) {
        assert (nacimiento != null) : "La fecha de nacimiento no es
válida";
        this.nacimiento = nacimiento;
        this.setEstado(EstadoAnimal.disponible);
        this.setRefugio(null);
        this.setAdopcion(null);
    }

    public Date getNacimiento() {
        return nacimiento;
    }

    public EstadoAnimal getEstado() {
        return estado;
    }

    protected void setEstado(EstadoAnimal estado) {
        assert (estado != null) : "El estado del animal no es
válido";
        this.estado = estado;
    }

    public Refugio getRefugio() {
        return refugio;
    }

    protected void setRefugio(Refugio refugio) {
        this.refugio = refugio;
    }

    public Adopcion getAdopcion() {
        return adopcion;
    }

    protected void setAdopcion(Adopcion adopcion) {
        this.adopcion = adopcion;
    }


    public String toString() {
        return "Animal nacido el: " + nacimiento.toString();
    }
}
```

```java
package Ejercicio1;

import java.util.Date;

public class Donacion {
   private double cantidad;
   private Date fecha;

   public Donacion(double cantidad, Date fecha) {
       this.setCantidad(cantidad);
       this.setFecha(fecha);
   }

   public double getCantidad() {
       return cantidad;
   }

   private void setCantidad(double cantidad) {
       assert (cantidad > 0) : "La cantidad a donar no puede ser
negativa o cero";
       this.cantidad = cantidad;
   }

   public Date getFecha() {
       return fecha;
   }

   private void setFecha(Date fecha) {
       assert (fecha != null) : "La fecha no es válida";
       this.fecha = fecha;
   }
}
```

```java
package Ejercicio1;

import java.util.*;

public class Donante extends Socio {
   private List<Donacion> donaciones;

   public Donante(Date registro, Refugio refugio) {
       super(registro, refugio);
       this.donaciones = new ArrayList<>();
   }

   public void donar(double cantidad) {
       assert (cantidad > 0) : "La cantidad donada debe ser
positiva";
```

```
        Date fechaActual = new Date();
        Donacion nuevaDonacion = new Donacion(cantidad,
fechaActual);
        donaciones.add(nuevaDonacion);
        this.getRefugio().addLiquidez(cantidad);
    }

    public Enumeration<Donacion> getDonaciones() {
        return Collections.enumeration(donaciones);
    }
}
```

```
package Ejercicio1;

public enum EstadoAnimal {
    disponible, adoptado, enTratamiento
}
```

```
package Ejercicio1;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Enumeration;
import java.util.List;

public class Refugio {
    private double liquidez;
    private List<Animal> animalesRegistrados;
    private List<Animal> animalesRefugiados;
    private List<Socio> socios;

    public Refugio(double liquidez) {
        this.setLiquidez(liquidez);
        this.animalesRegistrados = new ArrayList<>();
        this.animalesRefugiados = new ArrayList<>();
        this.socios = new ArrayList<>();
    }

    public void registrar(Animal animal) {
        assert (animal != null) : "El animal que se desea registrar
no es válido";
        assert (!animalesRegistrados.contains(animal)) : "El animal
que desea registrar ya está registrado";
        animalesRegistrados.add(animal);
        animal.setEstado(EstadoAnimal.disponible);
        animalesRefugiados.add(animal);
        animal.setRefugio(this);
    }
```

```
    public double getLiquidez() {
        return liquidez;
    }

    protected void setLiquidez(double liquidez) {
        assert (liquidez >= 0) : "La liquidez tiene que ser >=0";
        this.liquidez = liquidez;
    }

    protected void addLiquidez(double liquidez) {
        assert (liquidez > 0);
        this.liquidez += liquidez;
    }

    public Enumeration<Animal> getAnimalesRegistrados() {
        return Collections.enumeration(animalesRegistrados);
    }

    public Enumeration<Animal> getAnimalesRefugiados() {
        return Collections.enumeration(animalesRefugiados);
    }

    protected void removeAnimalRefugiado(Animal animal) {
        assert (animalesRefugiados.contains(animal)) : "El animal
no está en este refugio";
        assert (animal.getEstado() == EstadoAnimal.adoptado) : "El
animal no ha sido adoptado";
        animalesRefugiados.remove(animal);
        animal.setRefugio(null);
    }

    public Enumeration<Socio> getSocios() {
        return Collections.enumeration(socios);
    }

    protected void addSocio(Socio socio) {
        assert (socio != null) : "El socio introducido no es
válido";
        assert (!socios.contains(socio)) : "El socio introducido ya
está asociado";

        socios.add(socio);
    }

    protected void rmSocio(Socio socio) {
        assert (socio != null) : "El socio introducido no es
válido";
```

```
        assert (socios.contains(socio)) : "El socio introducido no
está asociado";
        socios.remove(socio);
    }
}
```

```
package Ejercicio1;

import java.util.Date;

public abstract class Socio {
    private Date registro;
    private Refugio refugio;

    public Socio(Date registro, Refugio refugio) {
        this.setRegistro(registro);
        this.setRefugio(refugio);
    }

    public Date getRegistro() {
        return this.registro;
    }

    private void setRegistro(Date registro) {
        assert (registro != null) : "La fecha de registro no puede
ser nula";
        this.registro = registro;
    }

    public Refugio getRefugio() {
        return refugio;
    }

    protected void setRefugio(Refugio refugio) {
        assert (refugio != null) : "El refugio no puede ser nulo";

        this.refugio = refugio;
        this.refugio.addSocio(this);
    }
}
```

```
package Ejercicio1;

import java.util.*;

public class Voluntario extends Socio {
    private List<Adopcion> adopciones;
```

```
    private List<Animal> animalesRegistradosPorVoluntario;

    public Voluntario(Date registro, Refugio refugio) {
        super(registro, refugio);
        adopciones = new ArrayList<>();
        animalesRegistradosPorVoluntario = new ArrayList<>();
    }

    public void registrar(Animal animal) {
        assert (animal != null) : "El animal no es válido";
        this.getRefugio().registrar(animal);
        animalesRegistradosPorVoluntario.add(animal);
    }

    public void tramitarAdopcion(Animal animal, Adoptante
adoptante) {
        assert (animal != null) : "El animal no es válido";
        assert (adoptante != null) : "El adoptante no es válido";
        assert (animal.getEstado() == EstadoAnimal.disponible) : "El
animal no está disponible para adopción";
        assert
(Collections.list(this.getRefugio().getAnimalesRefugiados()).conta
ins(animal)) : "El animal no pertenece a este refugio";

        Adopcion adopcion = new Adopcion(new Date(), animal,
adoptante, this);

        addAdopcion(adopcion);
        animal.setEstado(EstadoAnimal.adoptado);
        this.getRefugio().removeAnimalRefugiado(animal);
    }

    public Enumeration<Animal>
getAnimalesRegistradosPorVoluntario() {
        return
Collections.enumeration(animalesRegistradosPorVoluntario);
    }

    protected void addAdopcion(Adopcion adopcion) {
        assert (adopcion != null) : "La adopción no es válida";
        assert (!adopciones.contains(adopcion)) : "Ya existe esta
adopción";
        for (Adopcion ad : adopciones) {
            assert (ad.noDup(adopcion)) : "Ya existe una adopción
para este animal";
        }
        adopciones.add(adopcion);
    }
```

```
    public Enumeration<Adopcion> getAdopciones() {
        return Collections.enumeration(adopciones);
    }
}
```

CÓDIGO EJERCICIO 2

```
package Ejercicio2;

import java.util.Date;
import java.util.Objects;

public class Adopcion {
    private Date fecha;
    private Animal animal;
    private Adoptante adoptante;
    private Voluntario voluntario;

    public Adopcion(Date fecha, Animal animal, Adoptante adoptante,
Voluntario voluntario) {
        this.setFecha(fecha);
        this.setAnimal(animal);
        this.setAdoptante(adoptante);
        this.setVoluntario(voluntario);
    }

    public Date getFecha() {
        return fecha;
    }

    private void setFecha(Date fecha) {
        assert (fecha != null) : "La fecha no es válida";
        this.fecha = fecha;
    }

    public Animal getAnimal() {
        return animal;
    }

    protected void setAnimal(Animal animal) {
        assert (animal != null) : "El animal no es válido";
        this.animal = animal;
        this.animal.setAdopcion(this);
    }

    public Adoptante getAdoptante() {
```

```
        return adoptante;
    }

    protected void setAdoptante(Adoptante adoptante) {
        assert (adoptante != null) : "El adoptante no es válido";
        this.adoptante = adoptante;
    }

    public Voluntario getVoluntario() {
        return voluntario;
    }

    protected void setVoluntario(Voluntario voluntario) {
        assert (voluntario != null) : "El voluntario no es válido";
        this.voluntario = voluntario;
    }

    public boolean noDup(Adopcion otraAdopcion) {
        assert (otraAdopcion != null) : "La adopción no es válida";
        return !this.animal.equals(otraAdopcion.getAnimal());
    }

    @Override
    public boolean equals(Object o) {
        if (o == null || getClass() != o.getClass()) return false;
        Adopcion adopcion = (Adopcion) o;
        return Objects.equals(fecha, adopcion.fecha) &&
Objects.equals(animal, adopcion.animal) &&
Objects.equals(adoptante, adopcion.adoptante);
    }

    @Override
    public int hashCode() {
        return Objects.hash(fecha, animal, adoptante);
    }
}
```

```
package Ejercicio2;

import java.util.Enumeration;

public interface Adoptante {
    void adoptar(Animal animal, Voluntario voluntario);

    Enumeration<Adopcion> getAdopciones();

    void addAdopcion(Adopcion adopcion);
}
```

```java
package Ejercicio2;

import java.util.Date;

public class Animal {
    private final Date nacimiento;
    private EstadoAnimal estado;
    private Refugio refugio;
    private Adopcion adopcion;

    public Animal(Date nacimiento) {
        assert (nacimiento != null) : "La fecha de nacimiento no es
válida";
        this.nacimiento = nacimiento;
        this.setEstado(EstadoAnimal.disponible);
        this.setRefugio(null);
        this.setAdopcion(null);
    }

    public Date getNacimiento() {
        return nacimiento;
    }

    public EstadoAnimal getEstado() {
        return estado;
    }

    protected void setEstado(EstadoAnimal estado) {
        assert (estado != null) : "El estado del animal no es
válido";
        this.estado = estado;
    }

    public Refugio getRefugio() {
        return refugio;
    }

    protected void setRefugio(Refugio refugio) {
        this.refugio = refugio;
    }

    public Adopcion getAdopcion() {
        return adopcion;
    }

    protected void setAdopcion(Adopcion adopcion) {
        this.adopcion = adopcion;
```

```
    }

    public String toString() {
        return "Animal nacido el: " + nacimiento.toString();
    }
}
```

```
package Ejercicio2;

import java.util.Date;

public class Donacion {
    private double cantidad;
    private Date fecha;

    public Donacion(double cantidad, Date fecha) {
        this.setCantidad(cantidad);
        this.setFecha(fecha);
    }

    public double getCantidad() {
        return cantidad;
    }

    private void setCantidad(double cantidad) {
        assert (cantidad > 0) : "La cantidad a donar no puede ser
negativa o cero";
        this.cantidad = cantidad;
    }

    public Date getFecha() {
        return fecha;
    }

    private void setFecha(Date fecha) {
        assert (fecha != null) : "La fecha no es válida";
        this.fecha = fecha;
    }
}
```

```
package Ejercicio2;

import java.util.Enumeration;

public interface Donante {
    void donar(double cantidad);

    Enumeration<Donacion> getDonaciones();
```

```
}
```

```
package Ejercicio2;

public enum EstadoAnimal {
   disponible, adoptado, enTratamiento
}
```

```
package Ejercicio2;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Enumeration;
import java.util.List;

public class Refugio {
   private double liquidez;
   private List<Animal> animalesRegistrados;
   private List<Animal> animalesRefugiados;
   private List<Socio> socios;

   public Refugio(double liquidez) {
       this.setLiquidez(liquidez);
       this.animalesRegistrados = new ArrayList<>();
       this.animalesRefugiados = new ArrayList<>();
       this.socios = new ArrayList<>();
   }

   public void registrar(Animal animal) {
       assert (animal != null) : "El animal que se desea registrar
no es válido";
       assert (!animalesRegistrados.contains(animal)) : "El animal
que desea registrar ya está registrado";
       animalesRegistrados.add(animal);
       animal.setEstado(EstadoAnimal.disponible);
       animalesRefugiados.add(animal);
       animal.setRefugio(this);
   }

   public double getLiquidez() {
       return liquidez;
   }

   protected void setLiquidez(double liquidez) {
       assert (liquidez >= 0) : "La liquidez tiene que ser >=0";
       this.liquidez = liquidez;
   }
```

```
    protected void addLiquidez(double liquidez) {
        assert (liquidez > 0);
        this.liquidez += liquidez;
    }

    public Enumeration<Animal> getAnimalesRegistrados() {
        return Collections.enumeration(animalesRegistrados);
    }

    public Enumeration<Animal> getAnimalesRefugiados() {
        return Collections.enumeration(animalesRefugiados);
    }

    protected void removeAnimalRefugiado(Animal animal) {
        assert (animalesRefugiados.contains(animal)) : "El animal
no está en este refugio";
        assert (animal.getEstado() == EstadoAnimal.adoptado) : "El
animal no ha sido adoptado";
        animalesRefugiados.remove(animal);
        animal.setRefugio(null);
    }

    public Enumeration<Socio> getSocios() {
        return Collections.enumeration(socios);
    }

    protected void addSocio(Socio socio) {
        assert (socio != null) : "El socio introducido no es
válido";
        assert (!socios.contains(socio)) : "El socio introducido ya
está asociado";
        socios.add(socio);
    }

    protected void rmSocio(Socio socio) {
        assert (socio != null) : "El socio introducido no es
válido";
        assert (socios.contains(socio)) : "El socio introducido no
está asociado";
        socios.remove(socio);
    }
}
```

```
package Ejercicio2;

import java.util.*;
```

```java
public class RolAdoptante implements Adoptante {
   private Socio socio;
   private List<Adopcion> adopciones;

   public RolAdoptante(Socio socio) {
       setSocio(socio);
       this.adopciones = new ArrayList<>();
   }

   @Override
   public void adoptar(Animal animal, Voluntario voluntario) {
       assert (animal != null) : "El animal no es válido";
       assert (voluntario != null) : "El voluntario no es válido";

       Date fechaAdopcion = new Date();
       Adopcion adopcion = new Adopcion(fechaAdopcion, animal,
this, voluntario);

       assert (!adopciones.contains(adopcion)) : "Ya existe esta
adopción";

       for (Adopcion ad : adopciones) {
           assert (ad.noDup(adopcion)) : "Ya existe una adopción
para este animal";
       }

       addAdopcion(adopcion);
       animal.setAdopcion(adopcion);
       voluntario.tramitarAdopcion(animal, this);
   }

   @Override
   public Enumeration<Adopcion> getAdopciones() {
       return Collections.enumeration(adopciones);
   }

   @Override
   public void addAdopcion(Adopcion adopcion) {
       assert (adopcion != null) : "La adopción no es válida";
       assert (!adopciones.contains(adopcion)) : "Ya existe esta
adopción";
       for (Adopcion ad : adopciones) {
           assert (ad.noDup(adopcion)) : "Ya existe una adopción
para este animal";
       }
       adopciones.add(adopcion);
   }
```

```
    public Socio getSocio() {
        return socio;
    }

    public void setSocio(Socio socio) {
        assert (socio != null) : "El socio no es válido";
        this.socio = socio;
    }
}
```

```
package Ejercicio2;

import java.util.*;

public class RolDonante implements Donante {
    private Socio socio;
    private List<Donacion> donaciones;

    public RolDonante(Socio socio) {
        setSocio(socio);
        this.donaciones = new ArrayList<>();
    }

    @Override
    public void donar(double cantidad) {
        assert (cantidad > 0) : "La cantidad donada debe ser
positiva";
        Date fechaActual = new Date();
        Donacion nuevaDonacion = new Donacion(cantidad,
fechaActual);
        donaciones.add(nuevaDonacion);
        this.socio.getRefugio().addLiquidez(cantidad);
    }

    @Override
    public Enumeration<Donacion> getDonaciones() {
        return Collections.enumeration(donaciones);
    }

    public Socio getSocio() {
        return socio;
    }

    public void setSocio(Socio socio) {
        assert (socio != null) : "El socio no es válido";
        this.socio = socio;
    }
}
```

```java
package Ejercicio2;

import java.util.*;

public class RolVoluntario implements Voluntario {
   private Socio socio;
   private List<Adopcion> adopciones;
   private List<Animal> animalesRegistradosPorVoluntario;

   public RolVoluntario(Socio socio) {
       setSocio(socio);
       adopciones = new ArrayList<>();
       this.animalesRegistradosPorVoluntario = new ArrayList<>();
   }

   @Override
   public void registrar(Animal animal) {
       assert (animal != null) : "El animal no es válido";
       this.socio.getRefugio().registrar(animal);
       animalesRegistradosPorVoluntario.add(animal);
   }

   @Override
   public void tramitarAdopcion(Animal animal, Adoptante
adoptante) {
       assert (animal != null) : "El animal no es válido";
       assert (adoptante != null) : "El adoptante no es válido";
       assert (animal.getEstado() == EstadoAnimal.disponible) : "El
animal no está disponible para adopción";
       assert
(Collections.list(this.socio.getRefugio().getAnimalesRefugiados())
.contains(animal)) : "El animal no pertenece a este refugio";

       Adopcion adopcion = new Adopcion(new Date(), animal,
adoptante, this);

       addAdopcion(adopcion);
       animal.setEstado(EstadoAnimal.adoptado);
       this.socio.getRefugio().removeAnimalRefugiado(animal);
   }

   @Override
   public Enumeration<Animal>
getAnimalesRegistradosPorVoluntario() {
       return
Collections.enumeration(animalesRegistradosPorVoluntario);
   }
```

```
    @Override
    public void addAdopcion(Adopcion adopcion) {
        assert (adopcion != null) : "La adopción no es válida";
        assert (!adopciones.contains(adopcion)) : "Ya existe esta
adopción";
        for (Adopcion ad : adopciones) {
            assert (ad.noDup(adopcion)) : "Ya existe una adopción
para este animal";
        }
        adopciones.add(adopcion);
    }

    public Socio getSocio() {
        return socio;
    }

    public void setSocio(Socio socio) {
        assert (socio != null) : "El socio no es válido";
        this.socio = socio;
    }
}
```

```
package Ejercicio2;

import java.util.*;

public class Socio {

    private Date registro;
    private Refugio refugio;

    private Voluntario voluntario;
    private Donante donante;
    private Adoptante adoptante;

    public Socio(Date registro, Refugio refugio) {
        this.setRegistro(registro);
        this.setRefugio(refugio);
    }

    // Métodos comunes
    public Date getRegistro() {
        return this.registro;
    }

    private void setRegistro(Date registro) {
```

```
        assert (registro != null) : "La fecha de registro no puede
ser nula";
        this.registro = registro;
    }

    public Refugio getRefugio() {
        return refugio;
    }

    private void setRefugio(Refugio refugio) {
        assert (refugio != null) : "El refugio no puede ser nulo";
        this.refugio = refugio;
        this.refugio.addSocio(this);
    }

    public void addRolVoluntario() {
        assert (this.voluntario == null) : "El rol voluntario ya
está asignado";
        this.voluntario = new RolVoluntario(this);
    }

    public void addRolDonante() {
        assert (this.donante == null) : "El rol donante ya está
asignado";
        this.donante = new RolDonante(this);
    }

    public void addRolAdoptante() {
        assert (this.adoptante == null) : "El rol adoptante ya está
asignado";
        this.adoptante = new RolAdoptante(this);
    }

    public Voluntario getRolVoluntario() {
        return this.voluntario;
    }

    public Donante getRolDonante() {
        return this.donante;
    }

    public Adoptante getRolAdoptante() {
        return this.adoptante;
    }
}
```

```
package Ejercicio2;
```

```java
import java.util.Enumeration;

public interface Voluntario {
   void registrar(Animal animal);

   void tramitarAdopcion(Animal animal, Adoptante adoptante);

   Enumeration<Animal> getAnimalesRegistradosPorVoluntario();

   void addAdopcion(Adopcion adopcion);

}
```